Batch Application Events ................................................................. 5
  Section Level Events .................................................................. 5
    ➢ Advance Section .................................................................. 5
    ➢ After Last Object Printed .................................................. 6
    ➢ Before Level Break ............................................................... 6
    ➢ Do Balance Auditor ............................................................ 7
    ➢ Do Column Heading ............................................................. 7
    ➢ Do Section ........................................................................ 8
    ➢ Do Tabular Break ................................................................ 9
    ➢ End Brk Section .................................................................. 9
    ➢ End Lvl Brk Footer Section ............................................... 10
    ➢ End Lvl Brk Header Section ............................................... 10
    ➢ End Of Report .................................................................. 11
    ➢ End Section ...................................................................... 11
    ➢ End Variable .................................................................... 12
    ➢ Init Lvl Brk Footer Section .............................................. 12
    ➢ Init Lvl Brk Header Section .............................................. 13
    ➢ Initialize Section ................................................................ 13
    ➢ Refresh Section .................................................................. 14
    ➢ Suspend Section .................................................................. 15
  Page Footer Section Level Events ........................................ 15
    ➢ Initialize Page Footer ......................................................... 15
  Page Header Section Level Events ........................................ 16
    ➢ End Page Header ................................................................ 16
    ➢ Initialize Page Header ..................................................... 17
  Report Footer Section Level Events ........................................ 18
    ➢ End Report Footer .............................................................. 18
    ➢ Initialize Report Footer .................................................. 19
  Report Header Section Level Events ........................................ 19
    ➢ End Report Header ............................................................. 19
    ➢ Initialize Report Header .................................................. 20
  Variable Events ....................................................................... 21
    ➢ Clear Space ....................................................................... 21
    ➢ Column Inclusion ................................................................ 21
    ➢ Do Variable ...................................................................... 21
    ➢ Initialize Variable ........................................................... 22
    ➢ Skip Variable ..................................................................... 22
    ➢ Suspend Variable .............................................................. 23
  Constant Events ....................................................................... 24
    ➢ Do Constant ................................................................. 24
    ➢ End Constant ................................................................. 24
    ➢ Initialize Constant .......................................................... 25
    ➢ Skip Constant ................................................................... 25
Interactive Application Events .................................................. 26
J.D.Edwards Event Rules Quick Reference Guide

Control Events .............................................................................................................................................. 26

Button Control Events ................................................................................................................................. 26
  ➢ Button Clicked ....................................................................................................................................... 26
  ➢ Checkbox Selection Changed .................................................................................................................. 27
  ➢ Post Button Clicked ............................................................................................................................... 28

Edit Control Events ...................................................................................................................................... 30
  ➢ Control Is Entered ................................................................................................................................. 30
  ➢ Control Is Exited ................................................................................................................................... 31
  ➢ Control Is Exited And Changed – Asynch ............................................................................................. 32
  ➢ Control Is Exited And Changed – Inline ............................................................................................... 33
  ➢ Post Visual Assist Button Clicked .......................................................................................................... 34
  ➢ Text Clicked .......................................................................................................................................... 35
  ➢ Visual Assist Button Clicked .................................................................................................................. 35

Grid Control Events ..................................................................................................................................... 36
  ➢ Add Grid Record to DB – After .............................................................................................................. 36
  ➢ Add Grid Record to DB – Before ........................................................................................................... 37
  ➢ Add Last Entry Row to Grid .................................................................................................................... 38
  ➢ All Grid Recs Added to DB ..................................................................................................................... 38
  ➢ All grid Recs Deleted from DB ............................................................................................................... 39
  ➢ All Grid Recs Updated to DB .................................................................................................................. 40
  ➢ Delete Grid Rec from DB – After ........................................................................................................... 41
  ➢ Delete Grid Rec from DB – Before .......................................................................................................... 42
  ➢ Delete Grid Rec Verify – After ............................................................................................................... 43
  ➢ Delete Grid Rec Verify – Before ............................................................................................................ 44
  ➢ Double Click on Row Header .................................................................................................................. 45
  ➢ Get Custom Grid Row ............................................................................................................................ 46

If the Find button is pressed ........................................................................................................................ 47
  ➢ Grid Column Clicked ............................................................................................................................... 47
  ➢ Grid Record Is Fetched ............................................................................................................................ 48
  ➢ Kill Focus on Grid ................................................................................................................................. 48
  ➢ Row Is Exited ....................................................................................................................................... 48
  ➢ Row Is Exited And Changed – Asynch .................................................................................................... 49
  ➢ Row Is Exited And Changed – Inline ..................................................................................................... 51
  ➢ Set Focus on Grid ................................................................................................................................. 53

Hyperitem Control Events ............................................................................................................................ 53
  ➢ Button Clicked ..................................................................................................................................... 53
  ➢ Post Button Clicked ............................................................................................................................... 54

Parent Child Control Events .......................................................................................................................... 55
  ➢ All Grid Recs Deleted from DB ............................................................................................................... 55
  ➢ Delete Grid Rec from DB – After ........................................................................................................... 55
  ➢ Delete Grid Rec from DB – Before ......................................................................................................... 55
  ➢ Delete Grid Rec Verify – After ............................................................................................................. 55

Created by: Chintan Zatakia - 3 - JDE Technical Consultant
### Table Events
- Delete Grid Rec Verify – Before .......................................................... 55
- Double Click on Row Header ................................................................. 55
- Get Custom Tree Node ........................................................................... 55
- Kill Focus on Control ............................................................................... 56
- Node Is Collapsing .................................................................................... 56
- Node Is Expanding ..................................................................................... 57
- Set Focus on Control ................................................................................. 58
- Tree – Begin Drag Operation .................................................................... 58
- Tree – Cancel Drag Drop .......................................................................... 59
- Tree – Drag Over Node ............................................................................. 59
- Tree – End Drag Drop Operation .............................................................. 60
- Tree Node Is Deleted ................................................................................. 61
- Tree Node Level Changed ......................................................................... 62
- Tree Node Selection Changed ................................................................... 63

### Form Events
- Add Record to DB – After .......................................................................... 63
- Add Record to DB – Before ......................................................................... 64
- Clear Screen After Added .......................................................................... 65
- Clear Screen Before Add ............................................................................. 65
- Dialog Is Initialized .................................................................................... 66
- End Dialog .................................................................................................. 67
- Last Grid Record Has Been Read ............................................................... 68
- Post Dialog Is Initialized ........................................................................... 69
- Update Record to DB – After ..................................................................... 70
- Update Record to DB – Before ................................................................. 71
- Write Grid Line – After .............................................................................. 72
- Write Grid Line – Before ............................................................................. 73

### Tab Control
- Tab Page Is Initialized ............................................................................. 74
- Tab Page Is Selected .................................................................................. 74

### Table Conversion Events
- Data Changed ............................................................................................ 75
- Format Fetched .......................................................................................... 75
- Process Begin ............................................................................................. 76
- Process End ................................................................................................ 76
- Row Fetched ............................................................................................... 76
- Set Selection Append Flag ......................................................................... 77
- Set User Selection ....................................................................................... 78

### Table Events
- After Record Is Deleted ............................................................................... 80
- After Record Is Fetched ............................................................................... 81
- After Record Is Inserted .............................................................................. 82
- After Record Is Updated .............................................................................. 83
- Before Record Is Deleted ............................................................................ 83
Batch Application Events

Section Level Events

- **Advance Section**

Advance Section should be used if you need to perform processing on object before a fetch. This occurs each time you do a fetch from the database. If this section does not have a business view attached, then this event is processed once.

Available On

- Section Level event rule only.

Available Objects

- All section level and global objects.

Typical Usage

This event occurs every time there is a database Fetch.

Additional Notes

If processing needs to be done after the Fetch, use the EventBeforeTabularBreak event.

Processing Sequence

1. *InitSection*
2. *AdvanceSection*
3. ((DoTabularBreak), *AdvanceSection*), repeat
4. *DoSection*
After Last Object Printed

AfterLastObjectPrinted should be used to process information after a row has been output. This occurs after a row is printed to an output file.

Available On

- This is available at the Section Level event rules.

Typical Usage

This can be use to process information after a row has been output.

Processing Sequence

- Do Object
- End Object
- Last Object
- Advance Section

Before Level Break

BeforeLevelBreak should be used to do processing after a fetch, but before any level breaks are checked.

Available On

- This Event is available at Tabular Section Level event rules.

Typical Usage

This is used to process event rules at a stage where the records are fetched from the Database but nothing has been done to see if there should occur a level break.

Processing Sequence

1. Advance Section
2. Before Level Break
3. Advance Section
4. Tabular Break
Do Balance Auditor

Event Rules for this event process form interconnect information and generate the Balance Auditor link information to be displayed in the PDF file.

Available On

- This event is available at the section level of the tabular sections.

Typical Usage

Generally, the event rules for this event will be automatically generated through the Balance Auditor setup screens accessed through the section properties dialog. If this is the case, only form interconnect event rules will be written. This is the only event that form interconnect event rules are supported.

Additional Notes

While most of the event rules for this event will be automatically generated, the user does have the option to add their own custom logic. Users must realize that this event only processes for numeric variables in tabular sections. Although the event is accessed at the section level of event rule, it actually occurs during the object processing.

Processing Sequence

1. InitObject
2. DoObject
3. DoBalanceAuditor (only if the current object is numeric and the current section is tabular)
4. EndObject

Do Column Heading

DoColumnHeading event will happen once for every column header that is printed on a report. This applies to both columnar and tabular style sections. This also applies to when column headers are reprinted after a page break has occurred. Regardless of why the column headings are printing, this event will be executed.
Available On

- This event is available on column header objects only.

Typical Usage

Typically, this event would be used to conditionally change the value of the column header information to be printed.

Processing Sequence

- Initialize Section
- Advance Section
- Do Section
- DoColumnHeading (Once for every column header in the section)
- Initialize Object
- Do Object
- End Object

Do Section

DoSection occurs after Advance Section after values have been assigned to print out to a printer or an output file. This event occurs before any information for the current record is written to the PDF file. This event occurs before DoCell (if Tabular cells exist) and before DoVariable/DoConstant.

Available On

- Section level event rules.

Available Objects

- All Section Level Objects and Global Variables.

Typical Usage

Typical usage is if there is a need for processing after the values have been assigned to print to the PDF file/printer. Here the values to be printed are ready and so if there is a need for post processing, this is the place to do it.

Processing Sequence

1. Advance Section
2. Do Section
3. Do Object
4. End Object
5. Advance Section

- **Do Tabular Break**

DoTabularBreak occurs when any of the business view fields set as level breaks change. This is only valid for tabular sections. Use this to do processing that requires a change of values in any of the level break fields.

**Available On**

- This Event is available only in a Tabular Section with Tabular Breaks specified. Also this event is available only at the Section level event rules.

**Available Objects**

- The available objects are the business columns and RV fields.

**Typical Usage**

The typical usage would be to do any processing that requires a change of value in any of the Level Break fields.

**Additional Notes**

The Level Break fields are always one of the sequencing (sorting) fields.

**Processing Sequence**

1. Advance Section
2. Do Tabular Break
3. Do Section.

- **End Brk Section**

Event Rules attached to this event occur after a Level Break finishes. After the event has terminated, the Level Break Header begins processing.

**Available On**
The End Break Section Event is only visible to a section. It cannot be set for an object or the entire report.

Typical Usage

In order to process Event Rule logic immediately after a Level Break.

Processing Sequence

1. Level Break finishes.
2. Clears totals to accumulate new ones for next Break.
3. Processes Level Breaks Header.

➢ **End Lvl Brk Footer Section**

EndLvlBrkFooterSection should be used to do processing immediately after a level break footer.

Available On

• The End Level Break Footer Section Event is only visible to a section. It cannot be set for an object or the entire report.

Typical Usage

In order to process Event Rule logic immediately after a Level Break Footer.

Processing Sequence

1. Level Break Footer finishes.
3. Begin Header Section for next Section.

➢ **End Lvl Brk Header Section**

EndLvlBrkHeaderSection should be used to do processing immediately after a level break header.

Available On

• The End Level Break Header Section Event is only visible to a section. It cannot be set for an object or the entire report.

Typical Usage
In order to process Event Rule logic immediately after a Level Break Header.

Processing Sequence

1. Level Break Header finishes.
3. Begin processing objects for the body of section.

➤ **End Of Report**

EndOfReport sends INIT_PAGE_FOOTER and INIT_REPORT_FOOTER messages.

Available On

This event is currently unavailable.

Typical Usage

The typical use of this event is to update the values of global variables after all the level one sections are processed and before Page Footer and Report Footer are processed.

➤ **End Section**

EndSection occurs after a batch process has completed processing the last set of section values. Use this to do processing immediately after a section ends. This event is useful for last record and end of file procedures.

Available On

- The End Section Event is only visible to a section. It cannot be set for an object or the entire report.

Typical Usage

In order to process Event Rule logic immediately after a Section Ends.

Additional Notes

When a section is placed within a “parent” section, and is joined to the parent section by selected key values, it becomes a “child” section. The entire child section is processed for each associated set of parent
section values. An EndSection event occurs each time the child section is completed.

Processing Sequence

1. Section completes processing the objects within the section.
2. Clears totals to accumulate new ones for next Section.
3. Begin processing Section footer, if one exists.

- **End Variable**

EndVariable occurs immediately after the object has been processed, even if the object is invisible or suppressed.

Available On

- This event is only available to report objects. It is not applicable to the entire report or sections within the report.

Typical Usage

After an object has been processed, event rule logic may be processed.

Processing Sequence

2. Event Rule logic.

- **Init Lvl Brk Footer Section**

InitLvlBrkFooterSection is to be used to do processing immediately before a level break footer.

Available On

- The Initialize Level Break Footer Section Event is only visible to a section. It cannot be set for an object or the entire report.

Typical Usage

In order to process Event Rule logic immediately before a Level Break Footer.
Processing Sequence

1. Level break occurs.

- **Init Lvl Brk Header Section**

InitLvlBrkHeaderSection should be used to do processing immediately before a level break header.

**Available On**

- The Initialize Level Break Header Section Event is only visible to a section. It cannot be set for an object or the entire report.

**Typical Usage**

In order to process Event Rule logic immediately before a Level Break Header.

**Processing Sequence**

1. Level Break Header finishes.
3. The End Level Break Header tries to locate the header section, if it does, it sends the DO_BREAK_HEADER message for the header section; otherwise, it sends the INIT_BREAK_HEADER message for the next level.
4. Begin processing objects for the body of section.

- **Initialize Section**

InitializeSection occurs when a batch process encounters a section for the first time. Use this to do processing immediately before a section begins. This is useful for working with global variables or performing other preparatory procedures. For conditional sections, this event will be processed each time the section is called.

**Available On**

The End Section Event is only visible to a section. It cannot be set for an object or the entire report.
Typical Usage

In order to process Event Rule logic immediately before a Section Begins. This is a good point for working with global variables or performing other preparatory procedures.

Additional Notes

Initialize Section occurs before any of the structures or pointers related to the section have been established. Do not attempt to manipulate data which is only valid within the current section (since the pointers do not exist yet).

Processing Sequence

1. Previous section terminates.

- Refresh Section

The first time the UBE encounters a child section; it issues an initialize section event. Each subsequent time the child section is to be processed, the batch process uses Refresh Section. At this point, the internal structures and pointers for the child section have been established and the UBE is about to select a new group of records for the child section. This logic also works for the level break sections. use this to set the object values of level two sections based on the parent section. You can also use this event to reset or modify data selection and sequencing of the child section.

Available On

- Any level two section (child- level break header - level break footer-total section)

Typical Usage

The typical use of this event is to set the object values of level two sections based on the parent section. In addition, it is used for resetting/modifying data selection/sequencing of the child section.

Processing Sequence
1. Refresh Section
2. Adv Section

- **Suspend Section**

SuspendSection processes when an overflow page break occurs. For example, if the information printed does not fit in the space available on a page. This temporarily stops the section processing. Use this to do processing when a page break occurs.

Available On

- Section Level event rules.

Available Objects

- All Section Objects and Global Variables.

Typical Usage

To do processing when a Page Break occurs due to various reasons.

---

**Page Footer Section Level Events**

- **Initialize Page Footer**

InitializePageFooter occurs at the beginning of the report after any report header logic and before the page header section processes for the first time. Use this to initialize values to be printed in the current page footer section. These assignments typically depend on information processed so far on that page.

Available On

- This event is only available at the section level of a page footer section.

Available Objects

- Business view column fields are not allowed here because a page footer can not have a business view attached.

Typical Usage
The typical use for this event would be to initialize values that will be printed in the current page footer section. These assignments would typically depend on information that has been processed so far on that page.

Additional Notes

This event occurs once for every page break that occurs, and once at the end of the report.

Processing Sequence

**Scenario A**

- Page Break (due to a section overflowing a page)
- InitPageFooter
- InitPageHeader (for the next page, if applicable)
- DoSection (for the section that did not completely print)

**Scenario B**

- EndSection (for the last section of the report)
- InitPageFooter
- InitReportFooter (if applicable)

**Page Header Section Level Events**

- **End Page Header**

EndPageHeader after the page header finishes processing. Use this to do processing immediately after a page header.

Available On

- The End Page Header Event is only visible from a Page Header section.

Typical Usage

In order to process Event Rule logic immediately after a Page Header.

Additional Notes
This event can handle multiple Page Headers, as well as headers that reprint on a page break.

Processing Sequence

1. A Page Break occurs
2. Process Initial Page Header Section
3. Process Do Section for Page Header
4. Process End Page Header
5. Check for another page header, if one exists then repeat

➢ Initialize Page Header

InitializePageHeader occurs at the beginning of a report after any report header logic and before the page header section processes for the first time. It also processes every time a page break occurs. Use this to initialize values that cannot be set until after the report header logic executes. This is similar to Init Section for a normal group, columnar, or tabular section, except that it is only processed for a page header section.

Available On

- This event is only available at the section level of a page header section.

Available Objects

Business view column fields are not allowed here because a page header can not have a business view attached.

Typical Usage

The typical use for this event would be to initialize values that can not be set until after the report header logic has been executed. This event is similar to the InitSection event for a normal group, columnar or tabular section except that it is only processed for a page header section.

Additional Notes

This event occurs once for every page break that occurs, and once at the beginning of the report.
Processing Sequence

**Scenario A**

1. Page Break (due to a section overflowing a page)
2. InitPageFooter (if applicable)
3. InitPageHeader (for the next page)
4. DoSection (for the section that did not completely print)

**Scenario B**

1. InitReportHeader (if applicable)
2. DoSection (for report header, if applicable)
3. EndReportHeader (if applicable)
4. InitPageHeader
5. DoSection (for page header)
6. EndPageHeader

**Report Footer Section Level Events**

- **End Report Footer**

EndReportFooter occurs after the report footer processes. After processing finishes, the report terminates. Use this to do processing immediately after a report footer.

Available On

- The End Report Footer Event is only visible from a Report Footer section.

Typical Usage

In order to process Event Rule logic immediately after a Report Footer.

Additional Notes

This event can handle multiple Report Footers.

Processing Sequence

1. Process the *Initial Report Footer Section* event
2. Process the *Do Section for Report Footer* event
3. Process the *End Report Footer* event
4. Check for another report header, if one exists then repeat

- **Initialize Report Footer**

  InitializeReportFooter occurs once at the very end of a report after everything else processes and before the report footer prints. Use this to initialize values to print in the report footer.

  **Available On**

  - This event is only available at the section level of a report footer section.

  **Available Objects**

  - Business view column fields are not allowed here because report footers cannot have a business view attached.

  **Typical Usage**

  The typical use for this would be to initialize values to print in the report footer. Usually these values could not be calculated or assigned until the rest of the report has completed processing.

  **Processing Sequence**

  1. EndSection (for the last section of the report)
  2. InitPageFooter (if applicable)
  3. InitReportFooter

**Report Header Section Level Events**

- **End Report Header**

  EndReportHeader occurs after the report header processes. Then the report processes the page header for a report. Use this to do processing immediately after a report header.

  **Available On**

  - The End Report Header Event is only visible from a Report Header section.

  **Typical Usage**
In order to process Event Rule logic immediately after a Report Header.

Additional Notes

This event can handle multiple Report Headers.

Processing Sequence

1. Report processing begins
2. Process the Initial Report Header Section event
3. Process the Do Section for Report Header
4. Process the End Report Header event
5. Check for another report header, if one exists then repeat

 grote; Initialize Report Header

InitializeReportHeader processes once at the very beginning of the report before anything else in the report processes. Use this to initialize values at the beginning of a report. This is similar to Init Section for a normal group, columnar, or tabular section except that it only processes for a report header section.

Available On

- This event is only available at the section level of a report header section.

Available Objects

Business view column fields are not allowed here because a report header cannot have a business view attached.

Typical Usage

The typical use for this event would be to initialize values at the beginning of the report, before any other Event Rule logic is processed. This event is similar to the InitSection event for a normal group, columnar or tabular section except that it is only processed for a report header section.

Processing Sequence

1. InitReportHeader
2. DoSection (for report header)
3. EndReportHeader
4. InitPageHeader (if applicable)
5. DoSection (for page header) (if applicable)
6. EndPageHeader (if applicable)

**Variable Events**

- **Clear Space**

ClearSpace occurs when a section reaches its end. If there is a need to process any information at the time a Section is completed, this is the event at which to attach the ER.

**Available On**

- This event is available at the Section Level ER.

**Typical Usage**

This event is used to perform some tasks that need to be done after a section is completed.

**Processing Sequence**

1. End Section
2. Clear Space

- **Column Inclusion**

ColumnInclusion occurs right after a record is fetched.

- **Do Variable**

DoVariable occurs just before the font and color are selected for the report object and before the value of the object is translated into a printable string of characters and output to the page. This is your last chance to manipulate the value or display attributes of the object before output.

**Available On**

- This event is available at the Object Level.

**Typical Usage**
As this is the last chance to modify the value, typically it is used to reformat the value or update currency information.

Processing Sequence

1. Init Variable
2. Do Variable
3. End Variable

➢ **Initialize Variable**

InitializeVariable is invoked each time a report object, or variable, is to be processed.

Available On

- This event is only available to report objects. It is not applicable to the entire report or sections within the report.

Typical Usage

After an object has been processed, event rule logic may be processed. When invoking this process, the object’s position on the page has not yet been determined, so this would be a good time to perform processing that would affect the calculation of the object’s position.

Processing Sequence

2. Event Rule logic.

➢ **Skip Variable**

If an object will not fit on the current page, the UBE will issue a SkipVariable, which will cause the object to be bypassed until the next page begins processing.

Available On

- This event is only available at the object level.

Typical Usage
The typical use of this event is to change the value of an object at the page break.

Processing Sequence

- Skip Object
- Process Next Object (Init-Do-End Object)
- Adv Section

trusted Variable

If an object requires multiple lines to print, such as a long text string or column heading, and if only part of the object fits onto a page, then the UBE will issue a SuspendVariable, which causes processing of the object to be halted until the next page has been started. The value of the object has already been partially processed when this event happens, so this would be a risky time for Event Rules to manipulate that value.

Available On

This event is only available at the object level.

Available Objects

List field types (RV, VA, and so forth)

Typical Usage

Typically, this event can be used to modify the value at the page break. Most event rule processing is done before an object is suspended; therefore, use caution when running this event.

Processing Sequence

1. Suspend Object
2. Next Object (Init-Do-End Object)
3. Adv Section
Constant Events

- **Do Constant**

DoConstant occurs just before the font and color are selected for the report constant and before the value is output to the page. This is your last chance to manipulate the value or display attributes of the object before output.

Available On

This event is available at the Object Level for report constant objects only.

Available Objects

List field types (such as RV, VA).

Typical Usage

As this is the last chance to modify the value, it is typically used to change the value to be printed.

Processing Sequence

- Initialize Constant
- Do Constant
- End Constant

- **End Constant**

EndConstant occurs immediately after the constant has been processed.

Available On

This event is only available to report constants. It is not applicable to the entire report or sections within the report.

Available Objects

List field types (such as BC, RV, VA)

Typical Usage
After a constant has been processed, this event rule logic can be processed.

Processing Sequence

- Initialize Constant
- Do Constant
- End Constant

Event Rule logic

Process next object.

- **Initialize Constant**

InitializeConstant is invoked each time before a report constant is to be processed.

Available On

This event is only available to report constants. It is not applicable to the entire report or sections within the report.

Available Objects

List field types (such as, BC, RV, VA).

Typical Usage

This event occurs before the constant’s position has been calculated on the page.

Processing Sequence

- Initialize Constant
- Do Constant
- End Constant

- **Skip Constant**

If a constant object will not fit on the current page, the UBE will issue a SkipConstant message, which will cause the object to be bypassed until the next page begins processing.

Available On
This event is only available at the object level, for report constants only.

Available Objects

List field types (such as RV, VA).

Typical Usage

The typical use of this event is to change the value of a constant at the page break.

Processing Sequence

- Skip Constant
- Process Next Object (Init-Do-End Object or Constant)
- Adv Section

**Interactive Application Events**

**Control Events**

**Button Control Events**

- **Button Clicked**

Event Rules attached to ButtonClicked occur when a push button is clicked.

Available On

- All form types

Available Objects

- All objects are available

Typical Usage

This event used Form Interconnect to a related form. It uses Hide/Show Grid in a Headerless/Detail and Header/Detail form (for example, the Option Button). Hide button and group common event
rule logic in this event so that the same logic can be executed with a push of the button at various events. Used in conjunction with the Push Button system function.

Processing Sequence

1. When the button is clicked, the control receives a button clicked message.
2. The API JDERTDftPushButtonProc is called from Windows, where the ButtonClicked event rule will be called.

For the OK button, an error set will stop processing.

- **Checkbox Selection Changed**

Event Rules attached to CheckboxSelectionChanged occur when the check box is changed. Behaviors occur when the check box is clicked on and off.

To assign rules to the values of the check box, attach If/While statements to this event. This allows a behavior to occur only when the box is checked.

Available On

- Find/Browse
- Fix/Inspect
- Header Detail
- Headerless Detail
- Search/Select
- Parent/Child Browse

Available Objects

- All objects are available

Typical Usage

This event assigns values to Form Controls that have Y/N or 1/0 as allowed values. Use check boxes to indicate True/False or Yes/No values, and then set up rules that relate to those values.

Additional Notes
Check box events are similar to Radio Button events. See also If/While Clauses, Radio Button Control Events.

Testing the value of the checkbox on this event will yield the new value of the selection.

Processing Sequence

The event is executed when the main dialog procedure receives a button clicked notification. The API that calls the event rules is jdeMaintainDftCheckBox.

➢ **Post Button Clicked**

Event Rules attached to PostButtonClicked occur when a push-button is clicked immediately after the event rule for “ButtonClicked” is over.

Available On

- All form types

Available Objects

- All objects are available

Typical Usage

This event is used to check and set values of variables that may have changed during the button clicked event processing. In addition, the event is used in cases where there is a form interconnect on the button clicked event. The "Post button Clicked" event executes after the button clicked event has finished. Return values from the Form interconnect may be tested on this event to set flags. This may not always be the case if modeless processing is enabled.

Processing Sequence

1. When the button is clicked, the control receives a button clicked message.
2. The API JDERTDftPushButtonProc is called from Windows, where the "Button Clicked" event rule will be called.
3. Immediately after that, the “Post Button Clicked” event would be executed.
Radio Button Selection Changed

Event Rules attached to RadioButtonSelectionChanged occur when the radio button to which the event is attached is changed. Behaviors occur when the radio button is clicked.

Available On

- Find/Browse
- Fix/Inspect
- Header Detail
- Headerless Detail
- Search/Select
- Parent/Child Browse

Available Objects

- All objects are available.

Typical Usage

This event is used to display or toggle in a browse form as well as control the hiding and showing of field columns or grid column fields. Additionally, it is used to select from a range of choices, and then relate behaviors to the particular choice selected. Finally, it is used to indicate True/False or Yes/No values, and then set up behaviors that relate to those values.

Additional Notes

Every radio button is associated with a data item. Radio buttons with the same data item are referred to as a group of radio buttons. Within a group of radio buttons, only one may be selected at a time. Each radio button in a group of radio buttons requires a unique value. Radio button values are assigned at design time.

For example, assume there are three radio buttons all associated with data item CHAR (a character). The first radio button is labeled "Yes", the second "No", and the third "Maybe". Their respective values (as set up at design time) are Y, N, and M.

<table>
<thead>
<tr>
<th>FC Yes</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC No</td>
<td>N</td>
</tr>
</tbody>
</table>
FC Maybe | M

Only one radio button can be selected at a time, so the value of CHAR will be Y if Yes is selected, N if No is selected, or M if Maybe is selected.

For clarity of coding, always compare the value (Y, N, M) to the radio button FC being checked (FC Yes, FC No, FC Maybe).

Radio button events are very similar to Check Box events. See also If/While Clauses, Check Box Control Events.

Processing Sequence

1. A Windows message is received for button clicked which is tapped by the API jdeMaintainDftCheckBox. This API runs the event rule for Radio Button “Selection Changed”.

**Edit Control Events**

- **Control Is Entered**

ControlIsEntered executes each time a control receives focus.

**Available On**

- Find/Browse
- Fix/Inspect
- Header Detail
- Headerless Detail
- Search/Select
- Parent/Child Browse

**Available Objects**

- Applies to form controls only

**Typical Usage**

- Perform initializations related to the control receiving focus

**Additional Notes**
For performance reasons, this event should be used sparingly.

Processing Sequence

1. Run the Control Is Entered event rules.

Note: When focus is going from one control to another the Windows operating system sends the EN_SETFOCUS (Control Is Entered) for the control receiving focus before sending the EN_KILLFOCUS (Control is Exited) for the control loosing focus.

- **Control Is Exited**

ControlIsExited executes each time a control loses focus. It executes when focus is going to the Cancel button or to another dialog. This event is executed inline.

Available On

- Find/Browse
- Fix/Inspect
- Header Detail
- Headerless Detail
- Search/Select
- Parent/Child Browse

Available Objects

- All objects are available

Typical Usage

- Typically, this event is used with exited and changed events to maintain variables.

Additional Notes

For performance reasons should be used sparingly.

Processing Sequence

1. If the text from the control is different from the formatted data stored internally, store flags about modification
2. Clear any errors that were set on this control during the Control is Exited ER (from a previous call)
3. Run the Control is Exited ER

Note: When focus is going from one control to another the Windows operating system sends the EN_SETFOCUS (Control Is Entered) for the control receiving focus before sending the EN_KILLFOCUS (Control Is Exited) for the control loosing focus.

- **Control Is Exited And Changed – Asynch**

ControlIsExitedandChanged-Asynch executes each time a control loses focus if the text in the control has changed since a previous execution of this event. It is not run if the data in the control being left does not pass Data Type validation. It can execute even when focus is going to the cancel button or to another dialog. This event is executed anachronously.

Control is Exited and Changed-Asynch is put on the thread queue after Control is Exited and Changed-Inline event rule is executed.

**Available On**
- Find/Browse
- Fix/Inspect
- Header Detail
- Headerless Detail
- Search/Select
- Parent/Child Browse

**Available Objects**
- All objects are available.
- Using the Form Control being left in event rule gets the current value of the cell

**Typical Usage**
- May be used to run sizable amount of event rule logic and business functions
- This event should not depend on another ControlIsExitedandChanged-Asynch event because you cannot define the order in which the same two events are processed
- Do not use this event when validating filters for a Find

**Additional Notes**
This asynchronous event is executed even if the Control is Exited and Changed-Inline event rule sets an error. The execution of this event rule must finish before the form can be closed.

Processing Sequence

1. If the cell has not changed by typing since the last execution of this event, stop processing.
2. Perform Data Type validation for the data in the control. If there are errors, skip to 7.
3. Clear any errors that were set on this control during the Control is Exited and Changed-Asynch event rule (from a previous call).
4. Run the Control is Exited and Changed-Asynch event rule. If there are errors, skip to 7.
5. Perform Data Dictionary validation. If there are errors, skip to 7.
6. If there is an associated description, make that assignment.
7. If the control is set in error, a greater than filter is something other than a string, and the value in the field is not a wildcard, then clear the error.

Note: Most errors are cleared on filter fields due to the possibility of wildcards in the database query statement. Wildcards are not allowed in math numeric or dates, so those errors remain. Another idea where errors would be cleared is in filters that are not looking for an equal (exact) match. The idea here, for example, is that a string field set up as a greater than filter where “Elephant” is allowed should not set an error when “A” is entered, even if “A” is not a valid entry. It should accept “A” and find the “Elephant” record on the next Find.

Note: When focus is going from one control to another the Windows operating system sends the EN_SETFOCUS (Control Is Entered) for the control receiving focus before sending the EN_KILLFOCUS (Control is Exited) for the control losing focus.

- Control Is Exited And Changed – Inline

ControlIsExitedandChanged-Inline executes each time a control loses focus if the text in the control has changed since a previous execution of this event. It is not run if the data in the control being left does not pass Data Type validation. It can execute even when focus is going to the cancel button or to another dialog. Control is Exited and Changed - Inline executes after the Control is Exited event rule and before Control is Exited and Changed - Asynch event rule is put on the thread queue. This event is executed inline.
Available On

- Find/Browse
- Fix/Inspect
- Header Detail
- Headerless Detail
- Search/Select
- Parent/Child Browse

Available Objects

- All objects are available.
- Using the Form Control being left in event rule gets the current value of the cell unless it did not pass Data Type validation.

Typical Usage

- May be used to run event rule logic, which is required to be completed before running any other process.

Additional Notes

The execution of the Control is Exited and Changed - Asynch event occurs even if this inline event sets errors.

Processing Sequence

1. If the cell has not changed by typing or event rule since the last execution of this event, stop processing.
2. Clear any errors that were set on this control during the Control is Exited and Changed - Inline event rule (from a previous call).
3. Run the Control is Exited and Changed - Inline event rule

- **Post Visual Assist Button Clicked**

PostVisualAssistButtonClicked is processed after the visual assist form is closed.

Available On

- Any form type

Available Objects

- Available on any control that has a visual assist
Typical Usage

- Used to check and set values of variables that may have changed during the visual assist form processing.

Processing Sequence

1. This event is processed after the VisualAssistButtonClicked event.

- **Text Clicked**

   Text Clicked handles the functionality for static text that is clickable.

   Available On

   This event is available on all form types.

   Available Objects

   All field objects are available for this event.

   Typical Usage

   This event is typically used to transverse to another from or web site.

   Additional Notes

   None

   Processing Sequence

   If the text clicked is clickable, then when the user clicks on the text, the Text Clicked Event will be processed.

- **Visual Assist Button Clicked**

   Visual Assist Button Clicked is processed right after a visual assist button is clicked. It is before runtime brings up the default visual assist form.

   Available On

   - Any form
Available Objects

Available on any control that has a visual assist

Typical Usage

- Perform initialization or validation before the visual assist form opens.
- Suppress the default visual assist form and brings up an application-defined form using form interconnect.

Processing Sequence

1. When a visual assist button is clicked, runtime executes this event before it opens the default visual assist form.
2. After this event, runtime opens the default visual assist form if it is not suppressed.

Grid Control Events

- **Add Grid Record to DB – After**

Event Rules attached to AddGridRecordtoDB-After occur after a grid line has been added as a record to the database. Thus, the behaviors are not apparent on the current form and cannot affect the new record that has already been stored in the database.

Available On

- Headerless/Detail
- Header/Detail

Available Objects

- All objects are available

Typical Usage

Use this event to send information from a newly added record to another file (using a Business Purpose).

Additional Notes
This event differs from the corresponding event, Update Grid Record to DB, because the attached behavior occurs only after the addition of a new record to the database.

Processing Sequence

1. Perform database add.
2. Run AddGridRecordtoDB - After.

J.D. Edwards Standards

This event relates to grid records only. To work with form records, use the corresponding form events.

> Add Grid Record to DB – Before

Event Rules attached to AddGridRecordtoDB-Before occur after you press OK to add a grid line record to the database, but before the record is actually added.

Available On

- Headerless/Detail
- Header/Detail

Available Objects

- All objects are available.

Typical Usage

Use this when you want the behavior to affect the saved record. For example, use it to time-stamp a record entered from the grid into the database. If no records are to be added, use the SuppressAdd business function.

Additional Notes

This event differs from the corresponding event, UpdateGridRecordtoDB, because the attached behavior occurs relative to the addition of a new record to the database.

Processing Sequence
The BC’s are set up just as they are to be added to database. Run AddGridRecordtoDB - Before.

Add Last Entry Row to Grid

AddLastEntryRowtoGrid adds an entry row at the bottom of an update grid. Rules attached to this event affect the newly added row.

Available On

- Headerless/Detail
- Header/Detail

Available Objects

- All objects are available
- Grid columns are for the newly added row

Typical Usage

Use this event to assign line numbers and default values for new lines in Add and Update modes.

Additional Notes

To use this event, you must be working with an update grid. See also Grid Properties.

Processing Sequence

A key is pressed in the entry row or an assignment is made into the entry row.

Run the Add Last Entry Row to Grid event rule.

Physically add the new entry row to the grid.

All Grid Recs Added to DB

The Event Rules attached to AllGridRecsAddedtoDB occur after you add grid lines to the database. The behavior occurs after the records have been stored in the database. Thus, the behaviors are not apparent on the current screen and cannot manipulate the grid lines, since they have already been stored in the database.
Available On

- Headerless/Detail
- Header/Detail

Available Objects

- All objects are available.

Typical Usage

Use this event to perform behaviors based on newly added records to the database.

Additional Notes

This event is run even if there is only one grid record being added to the database.

This event runs after the AddGridRecordtoDB-After has run on the last added line in the grid. This event will execute only if the form is in Add mode. If the form is in Add mode, the All Grid Recs Updated to DB event will never execute. The form mode determines which All Grid Recs event will fire.

Processing Sequence

1. OK is pressed and all the grid records are added to the database.
2. Run the All Grid Recs Added to DB event rule.

- **All grid Recs Deleted from DB**

The event rules attached to AllGridRecsDeletedFromDB occur after grid lines and the corresponding database records have been deleted. The rules attached to this event are not apparent on the current form and cannot manipulate the grid lines or involve the deleted records.

Available On

- Find/Browse
- Headerless/Detail
- Header/Detail
- Parent/Child Browse

Available Objects
• All objects are available.

Typical Usage

Use this event to perform behaviors after deletion only.

Additional Notes

This event runs after DeleteGridRecfromDB-After runs on the last deleted line in the grid. This event runs even when only one grid record was selected for deletion.

Processing Sequence

For Find/Browse and Parent/Child Browse forms

1. Delete all selected grid records. Delete Grid Rec Verify - Before, Delete Grid Rec Verify - After, Delete Grid Rec from DB - Before, and Delete Grid Rec from DB - After are run for each selected row.
2. Run the All Grid Recs Deleted from DB event rule.

For Header/Detail and Headerless/Detail forms

1. When OK is pressed delete from the database all records that have been deleted from the grid. Delete Grid Rec from DB - Before, and Delete Grid Rec from DB - After are run for each record that was deleted from the grid.
2. Run the All Grid Recs Deleted from DB event rule.

➢ All Grid Recs Updated to DB

The event rules attached to AllGridRecsUpdatedtoDB occur after you update multiple grid lines, overwriting existing database records. The behavior occurs after the updated records have been stored in the database. The behaviors are not apparent on the current form and cannot manipulate the grid lines, since they have already been stored in the database.

Available On

• Headerless/Detail
• Header Detail
Available Objects

- All objects are available.

Typical Usage

Use this event to perform behaviors based on the updated records in the database.

Additional Notes

This event will execute even if only one grid record has been updated. This event will execute only if the form is in Update mode. If the form is in Update mode the All Grid Recs Added to DB event will not execute even if there were grid records added. In an update grid, it is possible to have some grid rows that need updated and some that need added. The form mode determines which All Grid Recs event will execute.

Processing Sequence

1. This event runs after all of the grid rows have been processed on the OK button when the form is in update mode.

   - Delete Grid Rec from DB – After

The event rules attached to DeleteGridRecfromDB-After occur during OK processing and after deletion from the database are complete. It does not occur if the delete was suppressed or the user did not confirm the delete.

Available On

- Find/Browse
- Headerless/Detail
- Header/Detail
- Parent/Child Browse

Available Objects

- All objects are available.

Typical Usage

This event can be used to delete related records from other tables (via business functions) once the deletion of the main record is complete.
• Use this to delete information in other tables and to maintain referential integrity.

Additional Notes

This event occurs after the delete from the database is complete, so it cannot be used to affect the database record or the grid row.

It is important to note that this event occurs immediately after the Delete Verify events for Find/Browse and Parent/Child forms, but it does not occur until the OK processing for Header/Detail and Headerless/Detail forms.

Processing Sequence

1. Delete the record from the database.
2. Run the Delete Grid Rec from DB - After event rule.

➢ **Delete Grid Rec from DB – Before**

The event rules attached to DeleteGridRecfromDB-Before occur after you press OK to delete a database record and after the Verify window is processed, but before the record is deleted.

If you cancel out of the Verify window, the behavior attached to this event will not occur.

Available On

• Find/Browse
• Headerless/Detail
• Header/Detail
• Parent/Child Browse

Available Objects

• All objects are available.
• Business view columns contain the record to be deleted.

Typical Usage

Use this event to update a related record with information from the record about to be deleted, or to log the deletion. The record still
exists in the database and the behavior attached to this event may use the record information, if needed.

Additional Notes

This event differs from the corresponding event, Delete Grid Rec Verify, because the attached behavior occurs relative to the deletion rather than the Verify window.

Processing Sequence

1. This event runs after the verify events but before the database delete. It is the last opportunity to call the Suppress Delete system function.

J. D. Edwards Standards

This event relates to grid records only.

➢ Delete Grid Rec Verify – After

The event rules attached to DeleteGridRecVerify-After after deletion is confirmed in the popup Verify window. If you cancel the Verify, the behavior attached to this event will not occur.

Available On

- Find/Browse
- Headerless/Detail
- Header/Detail
- Parent/Child Browse

Available Objects

- All objects are available.
- Grid columns contain the grid record to be deleted

Typical Usage

This event only runs if the user confirmed the delete, so it can be used for corresponding logic.

Additional Notes
This event differs from the corresponding Delete Grid Rec from DB event because the attached behavior occurs relative to the response entered on the Verify window rather the delete action.

Check for errors and suppress delete if needed.

Processing Sequence

1. This event runs after the Delete Confirmation dialog closes if the user selected Yes.

J. D. Edwards Standards

This event relates to grid records only.

➢ **Delete Grid Rec Verify – Before**

The event rules attached to DeleteGridRecVerify-Before occur after you press Delete for a selected grid record, but before the pop-up Verify window displays. Behaviors occur whether or not confirmation of the deletion is accepted or canceled in the Verify window.

Available On

- Find/Browse
- Headerless Detail
- Header/Detail
- Parent/Child

Available Objects

- All objects are available.
- Grid columns contain the grid record to be deleted

Typical Usage

This event typically provides an opportunity to suppress a delete before the confirmation dialog appears. If a suppress delete is requested in this event, the popup Verify window does not appear.

Additional Notes

This event differs from the corresponding DeleteGridRecordFromDB event because the attached behavior occurs relative to the appearance of the Verify window rather than the delete action
Processing Sequence

1. This event runs for every grid line selected at the time of the Delete button clicked. It is the first opportunity to call the Suppress Delete system function.

J.D. Edwards Standards

This event relates to grid records only.

- **Double Click on Row Header**

For DoubleClickRowHeader to be active, the row header must have been defined through grid properties in FDA. DoubleClickRowHeader is executed upon double clicking the row header.

Available On

- Search and Select
- Find/Browse
- Header/Detail
- Headerless/Detail

Available Objects

- All objects are available.

Typical Usage

This event is often used to exit to an attachments screen or to display media objects

Additional Notes

Double clicking on the row header causes this event to execute. Double clicking on the row causes a Press Button of the Select, if there is one, to execute.

Processing Sequence

This event will occur after Set Focus on Grid if focus was not on the grid prior to the double click.

**If focus was on the grid, but not on the row that was double clicked**
1. The cell that had focus will first execute the Leave Column events.
2. The row will execute the Leave Row events.
3. The double click will be processed.

If focus was on the grid row being double clicked

1. The Leave Column events will be processed for the cell loosing focus.
2. The double click will be processed.

- Get Custom Grid Row

Get Custom Grid Row executes when additional grid records are required in the grid. This event will execute when the Find button is clicked (first page of data is required), when the user scrolls down on the grid, and when the grid is resized. Get Custom Grid Row is used to allow a page at a time in custom filled grids. The event will continue to execute until a page of rows has been added or until the Continue Custom Fetch system function is not called. After the event has executed 1000 times without a grid row being added, a message will be written to the jde.log, but the event does not stop. This is a potential endless loop if the system function and event are not used properly.

Available On

- Find Browse
- Headerless Detail
- Header Detail
- Search Select

Available Objects

- All field types are allowed.

Typical Usage

This event is used for grids that are marked as No Fetch On Grid Business View. Typically, this event is used to insert a single custom grid row. The Find button ER will Open and Select from the tables/caches and the ER on Get Custom Grid Row will Fetch and insert a single row. After the row is inserted, the Continue Custom Fetch
system function will be called. If there are no more rows to insert, the
tables/caches will be closed and the system function is not called.

Additional Notes

It is possible for the form/fetch to terminate without retrieving all the
records. Assume there is more than a single page of data. The user
presses find, and the first page is fetched. The user then closes the
form. In this situation, the table/cache was opened in the Find ER, but
unless the table/cache is closed in End Dialog, it will remain open.
Similarly, when the user presses find, the first page is fetched, but
then the user changes the criteria and does another find. The
table/cache was opened in the Find ER, but unless the table/cache is
closed before the next find, it will still be open.

Processing Sequence

**If the Find button is pressed**

1. Find button ER
2. Delete all existing grid rows
3. Count number of existing grid rows
4. Do
5. Clear the Continue Custom Fetch system function flag
6. Get Custom Grid Row ER
7. Compare total number of grid rows to former to see if any were added
8. While more rows are required to have a page and the Continue Custom Fetch
   system function flag is set

**When more pages are required after the first page**

- Count number of existing grid rows
- Do
- Clear the Continue Custom Fetch system function flag
- Get Custom Grid Row ER
- Compare total number of grid rows to former to see if any were added
- While more rows are required to have a page and the Continue Custom Fetch
  system function flag is set

➤ **Grid Column Clicked**

Grid Column Clicked has been added to handle clickable grid columns.

Available On

- Available on all for types.

Created by: Chintan Zatakia - 47 - JDE Technical Consultant
Available Objects

- All field types are available.

Typical Usage

This event is used to traverse to another form or web site.

Processing Sequence

The engine will first check whether a particular grid column being selected is designated to be clickable. If it is, then the Grid Column Clicked event will be processed.

- **Grid Record Is Fetched**

GridRecordIsFetched is run each time that the interactive engine fetches a grid record from the database over the business view.

Available On

- Any form that has a grid.

Available Objects

- All objects are available
- Business view columns contain the record just fetched from the database
- Grid columns do not yet contain the fetched record

- **Kill Focus on Grid**

KillFocusonGrid is executed when the grid control loses focus.

Available On

- Find/Browse
- Header Detail
- Headerless Detail
- Search/Select
- Parent/Child Browse

Available Objects

- All objects are available.
Processing Sequence

If a cell had focus, the Leave Column events will execute then the Leave Row events will execute.

Run the Kill Focus on Grid event rule.

- **Row Is Exited**

RowisExited executes each time a grid row looses focus. It is run regardless of changes to the row. This event is executed inline.

Available On

- Find/Browse
- Header/Detail
- Headerless/Detail
- Search/Select

Available Objects

- All objects are available.
- Grid Columns contain the data of the row being left.

Typical Usage

This event is often used to apply conditional defaults to the row being left.

Processing Sequence

1. If the row was exited, but not the column, send a message to execute the Column is Exited events.
2. Clear any errors that were set on this row during the Row is Exited event rule (from a previous call).
3. Run the Row is Exited event rule.

- **Row Is Exited And Changed – Asynch**

RowisExitedandChanged-Asynch can execute each time a grid row looses focus. It is put on the thread queue immediately after the Row is Exited and Changed-Inline event rule is run. This event is executed anachronously.
For Full Process Grids (in grid properties, Process All Rows in Grid is marked), the event rule on Row is Exited and Changed-Asynch will execute when:

- Focus is moved to another row or to another control
- Okay is pressed, and one of the following is true:
  - Event rule has not yet been run for the row
  - A change has occurred on the row since a previous execution of this event

Note: These same rules apply to any row written to a grid, by means of a system function, with the “Updateable” flag as ‘YES’. If the “Updateable” flag is ‘NO’, then the Row is Exited event rule will never be run for that row.

For Read C Grids (in grid properties, Process All Rows in Grid is not marked), the event rule on Row is Exited and Changed - Asynch will execute when:

- A change has occurred to the grid row since a previous execution of this event and focus is moved to another row or to another control
- The OK button is clicked, event rule has not yet been run for the row, and the row was written to the grid, by means of a system function, with the “Updateable” flag as yes.

Available On

- Header Detail
- Headerless Detail

Available Objects

- All objects are available.
- Grid Columns contain the data of the row being left.

Typical Usage

May be used to run sizable amount of event rule logic, and business functions. This event should not depend on another RowIsExitedandChanged-Asynch event because you cannot define the order in which the same two events are processed.

Additional Notes
This event must complete before the form can be closed.

Processing Sequence

**For Full Process Grids and Read C Grids:**

1. If the cell has not changed by typing or event rule since the last execution of this event, skip to the last step in this section.
2. Clear any errors that were set on this row during the Row is Exited and Changed - Asynch event rule (from a previous call).
3. Run the Row is Exited and Changed - Asynch event rule.
4. Clear the flag so that this event is not run again until there is another change in the row.

**For All Grid Cells:**

1. Check to see if the data has already been set in error during validation, if it has then stop processing for this cell.
2. Check to see if the data completed validation through data changed processing, if it has then stop processing for this cell.
3. Load the default value if the cell is empty.
4. Check to see if the data is a required entry field, if it is, and it’s empty, set the field in error.
5. Perform Data Dictionary validation on the data in the cell.

- **Row Is Exited And Changed – Inline**

RowisExitedandChanged-Inline can execute each time a grid row looses focus. It will execute just before the Row is Exited and Changed - Asynch is . This event is executed inline.

For Full Process Grids (in grid properties, Process All Rows in Grid is marked), the event rule on Row is Exited and Changed - Inline will execute when:

- Focus is moved to another row or to another control
- Okay is pressed, and one of the following is true:
  1. Event rule has not yet been run for the row
  2. A change has occurred on the row since a previous execution of this event

Note: These same rules apply to any row written to the grid via a system function with the “Updateable” flag as ‘YES’. If the
“Updateable” flag is ‘NO’, then the Row is Exited and Changed - Inline event rule will never be run for that row.

For Read C Grids (in grid properties, Process All Rows in Grid is not marked), the event rule on Row is Exited and Changed - Asynch will execute when:

- A change has occurred to the grid row since a previous execution of this event and focus is moved to another row or to another control
- Okay is pressed, event rule has not yet been run for the row, and the row was written to the grid via a system function with the “Updateable” flag as yes.

Available On

- Header Detail
- Headerless Detail

Available Objects

- All objects are available.
- Grid Columns contain the data of the row being left.

Typical Usage

This event is used for processing that must complete before another row can be entered.

Processing Sequence

**For Full Process Grids and Read C Grids:**

1. If the cell has not changed by typing or event rule since the last execution of this event, stop processing
2. Clear any errors that were set on this row during the Row is Exited and Changed - Inline event rule (from a previous call).
3. Run the Row is Exited and Changed - Inline event rule.
4. Clear the flag so that this event is not run again until there is another change in the row.
Set Focus on Grid

SetFocusonGrid executes each time focus is moved from any control or any other dialog to the grid. It is important to note that this event will execute when switching between applications with the focus on the grid.

Available On

- Find/Browse
- Header Detail
- Headerless Detail
- Search/Select

Available Objects

- All objects are available. All event rules referring to Grid Columns will be performed with the data of, and on, the last line of the grid.

Typical Usage

Replace this text with a description of how the Event is typically used.

Processing Sequence

1. Run the Set Focus on Grid event rule.

Hyperitem Control Events

Button Clicked

Event Rules attached to ButtonClicked occur when a push button is clicked.

Available On

- All form types

Available Objects

- All objects are available
Typical Usage

This event used Form Interconnect to a related form. It uses Hide/Show Grid in a Headerless/Detail and Header/Detail form (for example, the Option Button). Hide button and group common event rule logic in this event so that the same logic can be executed with a push of the button at various events. Used in conjunction with the Push Button system function.

Processing Sequence

1. When the button is clicked, the control receives a button clicked message.
2. The API JDERTDftPushButtonProc is called from Windows, where the ButtonClicked event rule will be called.

For the OK button, an error set will stop processing.

- **Post Button Clicked**

Event Rules attached to PostButtonClicked occur when a push-button is clicked immediately after the event rule for "ButtonClicked" is over.

Available On

- All form types

Available Objects

- All objects are available

Typical Usage

This event is used to check and set values of variables that may have changed during the button clicked event processing. In addition, the event is used in cases where there is a form interconnect on the button clicked event. The "Post button Clicked" event executes after the button clicked event has finished. Return values from the Form interconnect may be tested on this event to set flags. This may not always be the case if modeless processing is enabled.

Processing Sequence
1. When the button is clicked, the control receives a button clicked message.
2. The API JDERTDftPushButtonProc is called from Windows, where the “Button Clicked” event rule will be called.
3. Immediately after that, the “Post Button Clicked” event would be executed.

**Parent Child Control Events**

- **All Grid Recs Deleted from DB**
  Please refer to the "AllGridRecsDeletedFromDB" event under the grid category.

- **Delete Grid Rec from DB – After**
  Please refer to the “DeleteGridRecFromDB-After” event under the grid category.

- **Delete Grid Rec from DB – Before**
  Please refer to the “DeleteGridRecFromDB-Before” event under the grid category.

- **Delete Grid Rec Verify – After**
  Please refer to the “DeleteGridRecVerify-After” event under the grid category.

- **Delete Grid Rec Verify – Before**
  Please refer to the “DeleteGridRecVerify-Before” event under the grid category.

- **Double Click on Row Header**
  Please refer to the “DoubleClickRowHeader” event under the grid category.

- **Get Custom Tree Node**
  GetCustomTreeNode executes when the Parent/Child form or the tree control needs more data to fill the tree. It is used for implementing page-at-a-time processing by the applications where the tree has to be loaded through system functions and event rules.
Available On

- Parent/Child Forms and Tree Controls.

Available Objects

- All form objects are available.

Typical Usage

This event can be used to implement page-at-a-time processing for custom fetched trees or parent/child forms. On this event, the application can use Table I/O to fetch data and use system functions to insert the fetched data into the tree. This event will be executed when the tree needs more data; the user either has clicked the Find button or has scrolled the tree to the bottom.

Additional Notes

There is a similar event for the forms with grids: "Get Custom Grid Row". The only difference is that "Get Custom Tree Node" is used for Parent/Child forms and Tree Controls; however, the "Get Custom Grid Row" event is used for Grids.

➢ **Kill Focus on Control**

KillFocusOnControl executes when the parent child control loses the focus.

Available On

- Parent Child forms

Available Objects

- All objects are available.

➢ **Node Is Collapsing**

NodeIsCollapsing executes when the user double-clicks on an open node or single clicks the “minus” sign of a node.

Available On

- Parent/Child Forms
Available Objects

- All objects are available.
- Grid Columns contain the data of the row that shows the detail of the node that is collapsing.

Typical Usage

The bitmap for the collapsing node can also be changed by calling “Set Tree Node Bitmap” with “Current Node” value for the “Node” parameter.

Additional Notes

This event is executed every time a node is collapsed.

The grid rows always reflect items that are relative to the current parent node in the tree, therefore, after the collapse of a node, the Grid is refreshed with items from the parent of the current node that is being collapsed. The selected row reflects the current node in the tree. The bitmap for the node is also changed to a default closed bitmap only if it is not overridden through the “Set Tree Node Bitmap” system function.

Node Is Expanding

NodeIsExpanding executes when the user double clicks on a node or single clicks on the “plus” sign of a node.

Available On

- Parent/Child Forms

Available Objects

- All objects are available.
- Grid Columns contain the data of the row that shows the detail of the node that is being expanded.

Typical Usage

The Application developer can use this event to add custom items under the presently expanding node. This can be achieved by first calling the system function “Suppress Fetch on Node Expand” and then
adding items by calling “Insert Grid Buffer Row”. The bitmap for the expanding node can also changed by calling “Set Tree Node Bitmap” with “Current Node” value for the “Node” parameter.

Additional Notes

This event is executed only once for each node that is expanded. If the expanding node has children, the tree node bitmap is changed to open bitmap; otherwise, the bitmap is changed to a "leaf" bitmap. Applications can use `SetTreeNodeBitmap` system functions to override the open or leaf bitmap.

- **Set Focus on Control**

  `SetFocusOnControl` executes when the parent child control receives the focus.

  **Available On**
  - Parent Child Form types.

  **Available Objects**
  - All objects are available.

- **Tree – Begin Drag Operation**

  `Tree-BeginDragOperation` executes when the user starts dragging a node in the tree. This event is only executed if the application developer has the drag and drop feature enabled.

  **Available On**
  - Parent Child Forms.

  **Available Objects**
  - All objects are available.
  - Grid Columns contain the data of the row that shows the detail of the node that is being dragged.

  **Typical Usage**

  Use this event to save the grid column values of the node (and row) that is being dragged into either grid buffers or variables so that they
can be used when the user has finished the drag and drop operation. The reason for saving off the grid column values is that when the user has dropped the node, the grid column values change to that of the node on which the dragged node has been dropped. The grid column values of the node that was dragged are replaced.

Additional Notes

The grid columns are also updated during the dragging process, when the user passes over a node in the tree. The node over which the user is moving gets highlighted and the grid columns now contain data for this node.

➢ **Tree – Cancel Drag Drop**

Tree-CancelDragDrop executes if the user drops (or releases) the dragged node when it is not on a valid node or outside the tree. This event is also executed if the user chooses the Cancel button from the popup menu that appears when the user uses the right mouse button during a mouse drag and drop operation.

Available On

- Parent Child Forms.

Available Objects

- All objects are available.
- Grid Columns contain the data of the row that shows the detail of the node that was initially dragged.

Typical Usage

The developer can use this event to undo steps they may have taken during the “Tree-BeginDragOperation”.

➢ **Tree – Drag Over Node**

Tree-DragOverNode executes when the user is in the process of dragging a node and is right above a node in the tree control. This node gets highlighted and the grid columns are updated with the values corresponding to this node. This event is only executed if the application developer has the drag and drop featured enabled.
Available On

- Parent Child Forms.

Available Objects

- All objects are available.
- Grid Columns contain the data of the row that contains the detail of the node over which the dragged node is passing.

Typical Usage

During this event, the application developer can inspect the grid column values and update the cursor to indicate if the user can or cannot drop the dragged node under the current node. Refer to the system function `Set Drag Cursor` for details on how to change the cursor.

- **Tree – End Drag Drop Operation**

Tree-EndDragDropOperation executes when the user has dropped the node that was being dragged. This event is only executed if the application developer has the drag and drop feature enabled.

Available On

- Parent Child Forms.

Available Objects

- All objects are available.
- Grid Columns contain the data of the row that shows the detail of the node on which the dragged node has been dropped.

Typical Usage

During this event, the application developer can update the tables to reflect the new structure of the tree using either business functions or Table IO. They can add a new record under the new parent and delete the record from the old parent if the operation was a MOVE rather than a COPY. To delete the dragged node from the tree, the developer can use the `DeleteGridRow` system function. To insert a node, the developer can use the `InsertGridBufferRow` system function.
Additional Notes

This event is not executed if the cursor has been set to NO DROP during the drag over event. Instead, a Cancel Drag Drop event is executed.

Processing Sequence

- If users make use of the left mouse button for the drag and drop operation, then this event is executed immediately after they release the left mouse button.
- If users make use of the right mouse button for the drag and drop operation, then a pop up menu comes up when they release the right mouse button. This popup menu has the following options:

  **Move Here**: This specifies that the user wants to move the dragged node. If the user selects this option then the Tree-EndDragDropOperation event will be executed.

  **Copy Here**: This specifies that the user wants to copy the dragged node. If the user selects this option then the Tree-EndDragDropOperation event will be executed.

Within this event, the application developer can check the system variable SV Drag_Mode to check if the operation was a COPY or a MOVE.

- **Tree Node Is Deleted**

TreeNodeIsDeleted executes when a tree node is deleted.

Available On

- Parent/Child Forms and Tree Controls.

Available Objects

- All form objects are available in this event.

Typical Usage
This event is executed when a tree node is deleted. It is used for the applications to clean up user data or table handles associated with the deleted node.

Additional Notes

When the user clicks the Find button or closes the form, the current tree is deleted. In this case, the event is executed for all the tree nodes.

- **TreeNode Level Changed**

TreeNodeLevelChanged executes when the user changes the selection of a tree node and the currently selected node is at a different level than that of the previously selected node. A typical example is, when the user moves from a parent level node to its child level node or vice versa.

Available On

- Parent Child Forms.

Available Objects

- All objects are available.

Typical Usage

This event could be used in a parent/child control with the grid side open to hide/show columns based on the current level of the tree node.

Additional Notes

This event is not executed if the tree node selection moves within the same level in the tree. If processing has to be for such cases, TreeNodeSelectionChanged event should be used.

Processing Sequence

1. This event does not prohibit the execution of TreeNodeSelectionChanged. In fact, TreeNodeLevelChanged is always executed before TreeNodeSelectionChanged incase both the events occur.
TreeNodeSelectionChanged executes when the user changes the selection of a tree node through either the mouse or the keyboard.

Available On

- Parent Child Forms.

Available Objects

- All objects are available.

Additional Notes

The TreeNodeLevelChanged event will be executed along with TreeNodeSelectionChanged event if the selection moves to a different level in the tree.

Form Events

- Add Record to DB – After

Event Rules attached to AddRecordtoDB-After occur after a form record has been added to the database. Any changes made to form controls or business view columns at this event point DO NOT get updated to the database.

Available On

- Fix/Inspect Form
- Header Detail Form
- Headerless Detail Form

Available Objects

- All objects are available.

Typical Usage
Use this event for processing after a record has been added to the database (For example, Display Total Amounts, Update Balance Files, PO History, Address Book Audit Information, and so forth).

Additional Notes

This event is only executed when adding a new record to the database. It is not executed during Update mode.

Processing Sequence

This event is run just after the add to the database has completed. It will not run if the database returns errors on the add.

- **Add Record to DB – Before**

Event Rules attached to AddRecordtoDB-Before occur after you press OK to add a form record to the database, but before the record is actually added to the database. For a Header Detail form, this event pertains to the header record, not the grid records.

Available On

- Fix/Inspect
- Header/Detail
- Available Objects
- All objects are available.

Typical Usage

Use this event when you want the behavior to affect the saved record. It is also a typical place to call the Suppress Add system function to stop the record from being saved.

Additional Notes

This event is only executed when adding a new record to the database. It is not executed during Update mode.

Processing Sequence

On the OK button in Add mode, if the SuppressAdd function has not been called, this event is run before the add takes place.
Clear Screen After Added

ClearScreenAfterAdded runs after the form has been cleared after an add has occurred. It differs from the ClearScreenBeforeAdd event because it is not run during the initial clear on entering the form.

Available On

- Fix/Inspect
- Header Detail
- Headerless Detail

Available Objects

- All objects are available.

Processing Sequence

1. Complete the addition of a database records
2. Clear all form items.
3. Run the ClearScreenBeforeAdd event
4. Run the PostDialogIsInitialized event
5. Run the ClearScreenAfterAdded event

Clear Screen Before Add

ClearScreenBeforeAdd runs any time a form is cleared before an add. This may occur if the form is entered in add mode, or if the fetch in update mode fails and the mode switches to add. The event runs after the form controls and business view columns are cleared or changed to the default value.

Available On

- Fix/Inspect
- Headerless/Detail
- Header/Detail

Available Objects

- All objects are available.

Typical Usage
J.D.Edwards Event Rules Quick Reference Guide

This event loads default values (other than data dictionary defaults) such as Branch/Plant, Company, and Company Currency.

Additional Notes

Controls flagged as Do Not Clear After Add are not cleared when this logic is performed. This event is also executed when an update/select is requested and the fetch fails.

Processing Sequence

As the form is opening:

1. Clear all form items
2. Run the ClearScreenBeforeAdd event
3. Run the PostDialogIsInitialized event

As OK is clicked:

1. Complete the addition of a database records
2. Clear all form items
3. Run the ClearScreenBeforeAdd event
4. Run the PostDialogIsInitialized event
5. Run the ClearScreenAfterAdded event

J.D. Edwards Standards

Errors issued at this event point cannot be removed at a later stage, forcing the user to exit the form using either the Exit or Cancel button.

Dialog Is Initialized

DialogIsInitialized executes each time a form is entered. It executes prior to the form being displayed on the screen.

Available On

- All form types

Available Objects

- All objects are available. Form interconnects have been loaded into corresponding business view columns and filter fields, if necessary.
Typical Usage

This event is often used for hiding and showing appropriate to interconnect values or processing options.

Processing Sequence

1. Memory space is allocated for each data element in the Business View (the header’s Business View, if a Header Detail form). If this form has been called previously, the allocated memory has been stored in the cache and will be reused.
2. Memory space is allocated for each control on the form. If the control is in the Business View, the control and the Business View will share memory space.
3. If there is a grid on this form, grid information is initialized.
   1. If this is a Header Detail form, memory space is allocated for each data element in the Business View for the grid. This information is cached, if possible.
   2. Memory space is allocated for each column in the grid. This information is cached, if possible.
4. Set the text on static fields and buttons. Button security is applied at this time.
5. Set up the form level and control level helps for this form.
6. Set up all the ER for this form. ER is not run at this point, but all preliminary validation occurs here, and TAM files are read, if necessary.
7. Initialize controls to their empty or default values.
8. Create the hypercontrol (Menu bar, toolbar, and button). Apply button security to hypercontrols.
9. Assign the Form Interconnect (FI) values to Business View (BC) values.
10. Run the Dialog is Initialized ER.

➢ **End Dialog**

DialogisInitialized is the reverse of the DialogisInitialized event and is always run just before a form is closed.

Available On

- All form types.

Available Objects
• All objects are available.

Typical Usage

Use this event to free memory allocated during entry in the form.

Additional Notes

This event is run before business view columns are copied to their corresponding form interconnections, so changes should therefore be made to the business view columns where there is automatic copying to propagate correctly.

Processing Sequence

1. This event runs.
2. Business view columns are copied to form interconnects.
3. The form closes.

➢ Last Grid Record Has Been Read

Event rules attached to LastGridRecordHasBeenRead occur after all grid records have been read into the grid.

Available On

• Find/Browse
• Header Detail
• Headerless Detail
• Search/Select

Available Objects

• All objects are available.

Typical Usage

Use this event to write total lines to the grid or display totals on the form level based on grid data. On Header/Detail forms, retrieve the last line number used in a transaction to allow addition of new lines.

Additional Notes

Errors issued at this event point cannot be removed at a later stage, forcing the user to exit the form using either the Exit or Cancel.
buttons. This event is processed even if no matching records are found.

Processing Sequence

1. Fetch matching records into grid until fetch fails.
2. Run this event. If records were fetched, previous event would be last *WriteGridLine-After*

➢ **Post Dialog Is Initialized**

PostDialogisInitialized executes after the Dialog is Initialized event. The processing sequence is dependent upon the form options and the mode of the form. This event is executed inline.

Available On

- Find/Browse
- Fix/Inspect
- Header Detail
- Headerless Detail
- Search/Select
- Parent/Child Browse

Available Objects

- All objects are available.
- Business view columns contain the record fetched, if any. In Copy Mode or Add Mode, the form interconnections have already been updated.

Typical Usage

To calculate the gross amounts, or modify the database fields before display on the form

Additional Notes

The sequencing of this event depends upon the mode and form options. When this event executes, a form record may or may not have been fetched by the interactive runtime engine. In some cases, it is important to know if a record has been fetched. If this information is required, the system function WasFormRecordFetched can be called.
Processing Sequence

- For forms entered in Update Mode when the form option "No Fetch On Form Business View" is cleared:
  1. Run the DialogisInitialized event.
  2. Fetch the record from database (this is the header record for transact forms).
  3. Run the PostDialogisInitialized event.
- For forms entered in Update Mode when the codegen option 'No Fetch On Form Business View' is selected:
  1. Run the DialogisInitialized event.
  2. Run the PostDialogisInitialized event.
- For forms entered in Copy Mode when the codegen option 'No Fetch on Form Business View' is cleared:
  1. Run the DialogisInitialized event.
  2. Fetch the record to be copied from the database (this is the header record for transact forms).
  3. If this is a transact form fetch the detail records (includes running grid record is fetched, etc).
  4. Clear the key fields.
  5. Run the ClearScreenBeforeAdd event.
  6. Run the PostDialogisInitialized event.
- For forms entered in Copy Mode when the codegen option "No Fetch on Form Business View" is selected:
  1. Run the DialogisInitialized event.
  2. Clear the key fields.
  3. Run the ClearScreenBeforeAdd event.
  4. Run the PostDialogisInitialized event.
- For forms entered in Add Mode when the codegen option "No Fetch On Form Business View" is selected:
  1. Run the DialogisInitialized event.
  2. Clear the fields.
  3. Run the ClearScreenBeforeAdd event.
  4. Run the PostDialogisInitialized event.

➤ **Update Record to DB – After**

Event Rules attached to UpdateRecordtoDB-After occur after a record has been updated to the database. Any changes made to form controls or business view columns at this event point DO NOT get updated to the database. This is a form level event so grid columns contain the values of the current row.

Available On
J.D.Edwards Event Rules Quick Reference Guide

- Fix/Inspect form and grid
- Header Detail form and grid
- Headerless Detail grid

Available Objects

- All objects are available.

Typical Usage

Use this event to perform processing after the record has been updated to the database, for example, Update Balance Files, PO History, or Address Book Audit Information.

Additional Notes

This event is only executed when updating an existing record to the database. It is not executed during Add mode. It applies to form records only.

Processing Sequence

This event runs as soon as the database update is complete as long as the database does not return in error.

- **Update Record to DB – Before**

Event rules attached to UpdateRecordtoDB-Before occur after you click OK to update a record that already exists in the database, but before the record is updated. For a Header/Detail form this event applies to the header and should not be confused with the grid event of a similar name.

Available On

- Fix/Inspect form and grid
- Header Detail form and grid
- Headerless Detail grid

Available Objects

- All objects are available.
Business view columns contain the record about to be updated. The business view columns should be used to alter the record about to be updated.

Typical Usage

Use this event to adjust values being saved to the database, to keep track of totals being saved, or suppress the update all together.

Additional Notes

This event is only executed when updating a record in the database. It is not executed during Add mode. This event updates form records only.

Processing Sequence

After all fields have been validated on OK processing in update mode, this event will run as long as the Suppress Update system function has not been called previously.

➤ **Write Grid Line – After**

WriteGridLine-After runs after a record retrieved from the database has been written out to the grid.

Available On

- Find/Browse
- Headerless Detail
- Header/Detail
- Search/Select
- Parent/Child Browse

Available Objects

- All objects are available.

Typical Usage

This event might be used for totaling or modifying grid column values. It can also be used to hide columns or change grid row attributes.

Processing Sequence
1. Run the WriteGridLine-Before event.
2. Run the AddFetchedRecordToGrid event.
3. Run the WriteGridLine-After event.

Additional Notes

Any changes to GC values during this event will result in an update to the database when OK is clicked.

➢ Write Grid Line – Before

WriteGridLine-Before executes for each grid line that is retrieved from the database after the grid column fields have been loaded with business columns but before the grid row has been written.

Available On

- Search and Select
- Find/Browse
- Headerless/Detail
- Header/Detail
- Parent/Child Browse

Available Objects

- All objects are available.

Typical Usage

- Write custom grid lines.
- Grid column manipulation (for example, Location formatting, Item Number formatting, and so forth).
- Any processing that needs to be performed with the grid column values should be performed here, such as business functions that fetch additional information, change presentation of information (Item Number), and so forth. If memory is allocated on a grid level (GENLNG), then this memory needs to be freed on the grid level as well.
- Defining colors and fonts for the grid.

Additional Notes
Errors issued at this event point cannot be removed at a later stage, forcing the user to exit the form using either the Exit or Cancel buttons.

Any changes to GC values during this event will not result in an update to the database when OK is clicked.

**Tab Control**

- **TabPage Is Initialized**

TabPageIsInitialized processes when a page is activated in the tab control for the first time.

Available On

- Each Page of the Tab Control

Available Objects

- All Objects are available.

Typical Usage

This event can be used to perform one time initialization for any tab page.

Additional Notes

This event is executed only once for each page, whereas the TabPageIsSelected event is processed each time a page is activated.

- **TabPage Is Selected**

TabPageIsSelected processes each time a page is activated in the tab control.

Available On

- Each Page of the Tab Control
Available Objects

- All Objects are available.

Additional Notes

This event is executed each time a page is activated, whereas the \textit{TabPageIsInitialized} event is processed only once for each page.

Processing Sequence

The first time a tab page is activated, this event is always executed after the \textit{TabPageIsInitialized} event.

**Table Conversion Events**

- **Data Changed**

  DataChanged happens when a sequenced input column’s data changes to a new value. Data Changed events are generated from the columns selected for sequencing. Their format is “Short Name” Data Changed. For example, if an input table is sequenced on System code then the corresponding event would have the form “SY Data Changed”.

  Typical Usage

  This event is used to output total information when data changes.

  Processing Sequence

  This event happens when the value of a sequenced column changes.

- **Format Fetched**

  FormatFetched happens each time that a format is read from the input table.

  Typical Usage

  This event is used to setup relationships (mappings) among the input formats and output tables.

  Processing Sequence
This event happens each time that a format record is read from the input table

- **Process Begin**

ProcessBegin allows for the set up of variables, and constant/literal data, which will exist while the Table Conversion, is executing. ProcessBegin is the first event of a Table Conversion.

Typical Usage

This event is used to initialize variables and literals.

Processing Sequence

First Event run for a Table Conversion

- **Process End**

ProcessEnd lends itself well to “last record” or “end of file” procedures. After Table Conversion has completed processing all of the mappings, ProcessEnd occurs.

Typical Usage

This event is used to write total, process validation, etc.

Processing Sequence

Last Event run for a Table Conversion

- **Row Fetched**

RowFetched happens every time a record is read from the input table.

Typical Usage

This event is used to setup relationships (mappings) between the input and output tables.

Processing Sequence

Row Fetched Event happens each time that a record is read from the input table.
Set Selection Append Flag

SetSelectionAppendFlag sets a flag to determine whether selection criteria added by the system function SetUserSelection will be appended to or replace the existing selection criteria for that section.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>I/O</th>
<th>Required</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append Flag?</td>
<td>YES or a NO. YES means that selection criteria will be appended to the current criteria.</td>
<td>I</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

Return Values

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Special Handling Instructions and Prerequisites

Because selection criteria will only typically be used within the initialization of a section, the call to this system function would probably want to be placed in the InitSection event. This function will also need to be called before the call to SetUserSelection if the criteria to be added through that system function must be appended.

Related Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
<th>Source Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearUserSelection</td>
<td>Removes existing data selection information for the current section.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetUserSelection</td>
<td>Adds or replaces data selection information for the current section.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClearUserSequence</td>
<td>Removes existing data sequencing information for the current section.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetSequenceAppendedFlag</td>
<td>Sets a flag used by SetUserSequence. The</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
flag determines whether to replace or append the existing data sequencing information.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>I/O</th>
<th>Required</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item1</td>
<td>This value is a Business View Column.</td>
<td>I Y</td>
<td></td>
<td>This field must be a business view, because it is this value that will be returned from a fetch. Therefore, this value must be compared to determine whether it fits the data selection criteria.</td>
</tr>
<tr>
<td>Comparison</td>
<td>The values for this parameter can be Equals, Not Equals, Greater Than, Less Than, Greater Than Or Equal To, or Less Than Or Equal To.</td>
<td>I Y</td>
<td></td>
<td>This field determines the type of comparison to be used between Item1 and Item2 when determining whether to include values.</td>
</tr>
<tr>
<td>Item2</td>
<td>This field can be any literal value</td>
<td>I Y</td>
<td></td>
<td>This field can be a business view column,</td>
</tr>
</tbody>
</table>

**Set User Selection**

SetUserSelection provides users with a method of conditionally modifying data selection for a section. This system function allows users to utilize ER to determine whether or not to make changes. This works in conjunction with the data selection for a section set up through RDA. Users should call SetSelectionAppendFlag prior to calling this function to determine whether to replace or append to the existing data selection information for this section.
| or object that can be compared with Item1. | constant, report variable, report constant, system value, system variable, or a literal value. The value of this field will be compared with Item1 when a fetch is done, to determine if the value returned from a fetch adheres to the data selection. |
| And/Or | This can be three values: And, Or, or None. | I | Y | ‘None’ is used when this is to be the first line of the selection criteria. The other two values should be used for subsequent lines. The value will specify whether this line should be treated as an ‘AND’ condition or an ‘OR’ condition. |

**Return Values**

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Special Handling Instructions and Prerequisites**

Because selection criteria will only typically be used within the initialization of a section, the call to this system function would probably want to be placed in the InitSection event.

**Related Functions**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
<th>Source Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearUserSelection</td>
<td>Removes existing data selection information for the current section.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetSelectionAppendFlag</td>
<td>Sets a flag used by</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetUserSelection</td>
<td>The flag determines whether to replace or append to the existing data selection information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClearUserSequence</td>
<td>Removes existing data sequencing information for the current section.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetSequenceAppnedFlag</td>
<td>Sets a flag used by SetUserSequence. The flag determines whether to replace or append the existing data sequencing information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetUserSequence</td>
<td>Adds or replaces data sequencing information for the current section.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table Events

- **After Record Is Deleted**

  AfterRecordIsDeleted executes after JDEBASE middleware deletes a record from the table.

  **Available On**
  - All OneWorld defined tables

  **Available Objects**
  - System Values (SL)
  - Table Columns (TK)
  - User Defined Variables (VA)

  **Typical Usage**

  This event is used for ensuring database integrity across different relational tables in the database. This event can also be used for sending messages to supervisors or for auditing Functions. For
example, if an item is deleted from the Item Master (F4101) there is a corresponding delete from the Bulk Item Master (F41011) for database integrity.

Additional Notes

Attach logic for deleting related table records on this event.

Processing Sequence

1. This event is executed followed by the actual deletion from the database by the middleware, then based on the whether the deletion was successful or not, this event will execute.

➢ **After Record Is Fetched**

AfterRecordIsFetched executes after record is fetched from the database and the JDEBASE middleware has it in the buffer, but not passed on to the application.

Available On

- All OneWorld defined tables

Available Objects

- System Values (SL)
- Table Columns (TK)
- User Defined Variables (VA)

Typical Usage

This event is processed after a Fetch has been done from the table and the columns are in the buffer. This is a good time to work with the values that will be passed on the application.

An example of this would be in the F0901 table where on this event Table ER logic will test to see if the user has a language preference set. If not, it will use the default account descriptions in the F0901 file. If the user does have a language preference, the ER will test to see if an alternative account description exists for the set language preference. If an alternative description does exist in the F0901D file for the set language preference, it will be used instead of the default
description. Otherwise, the default description will be used from the F0901 file.

Additional Notes

The OneWorld application will not distinguish between any TK values changed in the buffer, for example, on this event, versus a value coming from the database table.

Processing Sequence

The Before Record is Fetched event is executed followed by the actual fetch from the database by the middleware, and then based on the whether the fetch was successful or not, this event will execute.

- **After Record Is Inserted**

AfterRecordIsInserted executes after JDEBASE middleware inserts a record into the table.

Available On

- All OneWorld defined tables

Available Objects

- System Values (SL)
- Table Columns (TK)
- User Defined Variables (VA)

Typical Usage

This event is processed after an Insert has been done in the table. It is typically used for audit, sending informational messages, or inserting records in other tables for database integrity. For example, in the Order Approval Audit (F4818) Table, a Send Message system function is called to send a message to a designated supervisor.

Additional Notes

On this event, attach logic for inserting records in related tables.

Processing Sequence

1. This event will not execute If the insert failed.
After Record Is Updated

AfterRecordIsUpdated executes after a record is updated in the database by the JDEBASE middleware.

Available On

- All OneWorld defined tables

Available Objects

- System Values (SL)
- Table Columns (TK)
- User Defined Variables (VA)

Typical Usage

The logic on this event is executed after the middleware has updated the record in the database. It is typically used for audit, sending informational messages, or updating records in other tables for database integrity. For example, in the Order Approval Audit Table (F4818), a Send Message system function is called to send a message to a designated supervisor.

Additional Notes

On this process, attach logic for updating related tables.

Processing Sequence

This event will not execute if the update failed.

Before Record Is Deleted

BeforeRecordIsDeleted is triggered when an application calls the JDEBASE to Delete a record in this table.

Available On

- All OneWorld defined tables

Available Objects

- System Values (SL)
- Table Columns (TK)
• User Defined Variables (VA)

Typical Usage

This event rule is called before a record is deleted in the table. Is used for audit, and sending informational messages. For example, in the Shipment Header Table (F4215), a delete to multiple other tables, such as the Routing Steps Table (F4941), is done.

Additional Notes

Do not attach logic to delete related tables on this event as the Delete to this table might fail while other related tables would have been deleted. Attach logic for that process on the After Record is Deleted event.

Processing Sequence

This event executes after the application submits a request to the middleware to delete a record from the database. This event executes if the middleware is unable to delete this record. The After Record is Deleted event will not take place if the delete is unsuccessful.

➢ Before Record Is Fetched

When a OneWorld application requests the JDEBASE middleware to fetch a record from the database.

Available On

• All OneWorld defined tables

Available Objects

• System Values (SL)
• Table Columns (TK)
• User Defined Variables (VA)

Typical Usage

This event rule is called before a record is fetched from the table. It is used for sending informational messages or checking certain conditions that might affect the fetch.

Additional Notes
The request from the application to the middleware for the fetch has not yet been processed on this event. The actual fetch happens next and then populates the buffer.

Processing Sequence

After the application submits a request to the middleware to fetch a record from the database this event will be executed. If the middleware is unable to fetch this record, this event will still execute. The After Record is Fetched event will not take place if the fetch was unsuccessful.

- **Before Record Is Inserted**

After an application has issued a request for a record to be inserted and before the JDEBASE middleware actually inserts the record into the database.

Available On

- All OneWorld defined tables

Available Objects

- System Values (SL)
- Table Columns (TK)
- User Defined Variables (VA)

Typical Usage

This event rule is called before a record is inserted in the table. It is typically used for audit, sending informational messages, or inserting or updating records in other tables for database integrity or message files.

This event can also be used to stamp audit information (for example, date, time, user ID, or program ID) on records being Inserted in the database.

For example, in the Held Orders Table (F4209), a message is sent to users depending on a certain code in the table buffer and the PPAT Message (F01131) table is updated.

Additional Notes
Do not attach logic to insert records in related tables on this event as the insert to this table might fail while other related tables would have had records inserted whether or not the actual Insert to this table was successful or not. Attach logic for Inserting records to related tables on the *After Record is Inserted* event.

TK column values in the buffer might be changed at this point, which will affect the table update.

**Processing Sequence**

1. The event is executed after the application submits a request to the middleware to Insert a record into the database. This event will execute even if the middleware is unable to insert this record. The *After Record is Inserted* event will not take place if the Insert was unsuccessful.

- **Before Record Is Updated**

    BeforeRecordIsUpdated is triggered when an application calls the JDEBASE middleware to update a record in the database.

    **Available On**
    - All OneWorld defined tables.

    **Available Objects**
    - System Values (SL)
    - Table Columns (TK)
    - User Defined Variables (VA)

    **Typical Usage**

    This event rule is called before a record is updated in the table. It is typically used for audit, sending informational messages, or updating records in other tables for database integrity.

    This event can also be used for retrieving and writing records to the Cache.

    This event can also be used to stamp audit info (for example, date, time, user ID, or program ID) on records being updated in the database.
For example, in the Employee Master (F060116) table, an update is tracked and all columns in the table being updated are tracked for audit Functions. This is the point when they record the information when it is being updated. Logic attached to the After record is Fetched event, would have captured the initial picture of the record in the Cache and can be used at this event for comparison.

Additional Notes

Do not attach logic to update related tables on this event as the Update to this table might fail while other related tables would have been updated. Attach logic for that process on the After Record is Updated event.

Processing Sequence

This event executes after the application submits a request to the middleware to update a record from the database. This event will still execute if the middleware is unable to update this record. The After Record is Updated event will not take place if the Update was unsuccessful.

➤ **Currency Conversion Is On**

CurrencyConversionIsOn is used to alter the currency decimal in a currency related amount field.

Available On

- All OneWorld defined tables

Available Objects

- System Values (SL)
- Table Columns (TK)
- User Defined Variables (VA)

Typical Usage

Based upon the action of the database request, this event will call ‘After Record is Fetched’ on Inquiry mode and ‘Before Record is Added’ on Add/Update mode. Internally, these two events are executed by the *Currency Conversion is On* event, user does not need to enter logic in these two events.
Use this event to retrieve the currency code and currency decimal to a currency amount field.

For example, in the Account Ledge Table (F0911) currency decimal and currency code are needed for all the currency amount fields. A business function with special currency logic is used to retrieve the currency information and pass it back to the database. This currency information is stored within the amount field structure.

Additional Notes

Refer to the OneWorld Currency Processing document for more information.

Processing Sequence

Called whenever Currency is turned on in OneWorld.